Knowledge representation

lecture 2 second part, lecture 3 first part:

Schemaless databases,

RDF and RDFS.

T. Tammet

Alternatives to relational databases

A varied landscape of technologies,
part of which are under a NoSQL umbrella name:

- Network databases
- Graph databases
- Document databases
- Key-value databases
- Object-relational mapping
- Main memory databases
- XML databases
- RDF, Sparql, semantic web
- Google Bigtable and MapReduce framework

....

Triplets: an obvious idea to Implement schemaless databases

Each row with N cols is represented as N rows of three columns like

client_1 name „John Brown“

client_1 balance 200

Columns are called sometimes as

- Row/Object id Column name Value
- Object Property Value
- Subject Predicate Object

Similar to key-value

Object Property Value

can be combined to a key-value

Object:Property Value

What do we lose with the key-value representation?

Schema-less is often inevitable

Read data from numerous sources, aggregate in our own database:

- We have no control over foreign data
- Our understanding of foreign data changes
- Our data sources change

There is one „schemaless“ standard, but beware

There is a wide range of schemaless databases, but most of them are basically API-s or have proprietary query languages: no real standards.

However, there is one standard **- RDF (resource description framework)** – which is not really loved.

- Developed and pushed by W3C
- Cornerstone of the semantic web project
- Large number of systems supporting
- A lot of tools

RDF: triple not really a triple

Object Property **Value** **Valuetype**

With valuetype normally being either:

- One of xml schema datatypes
- Global id: URI
- Local id

RDF: some restrictions

Object Property **Value** Valuetype

Object, Property, Valuetype: URI-s

Value: URI or literal value

Many representation syntaxes

- RDF/XML
- RDFa
- N3
- N-triples
- Turtle
- ….

Example in Turtle syntax

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  ex:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

How to add metadata to a row?

Like timestamp, changer, row id, status etc etc?

Unpleasant answer: **reification**

How does reification work?

We have

personid:12 salary 20000

Want to add timestamp and entering person?

The reification way

From

personid:12 salary 2000 + timestamp etc

To

datarow:10001	subject	personid:12
datarow:10001	predicate	salary
datarow:10001	object	2000
datarow:10001	timestamp	2009-10-20 13:45
datarow:10001	modifier	personid:345

From the relational db ...

One row, N cols in the relational db

First, get N rows of four cols in RDF

Second, get (N*3)+X rows of four cols after reification

N → 12*N (is 12 a correct number here??)

Problem with containers

RDF provides a *container vocabulary* consisting of three predefined types (together with some associated predefined properties).

A *container* is a resource that contains things. The contained things are called *members*. The members of a container may be resources (including blank nodes) or literals. RDF defines three types of containers:

rdf:Bag

rdf:Seq

rdf:Alt

Problem with containers

Containers are mostly fake:

- Containers have no real semantics in RDF
- Container semantics would make queries very hard.

Problem with local id-s

Different object id-s:

- Global URI-s.
  - These are fine.
- Local “blank nodes”.
  - Their semantics/use in the RDF spec is broken: creates unnecessary problems.

Storage of RDF in a relational db

How to keep predicate, subject, valuetype URI-s:

Two main options:

- keep a separate table for unique strings
- use numeric string id-s in pred,subject,valuetype

Storage of rdf in a relational db

Storing value? Can be int, float, string, URI, ...

Several ways, none really pleasant:

- Encode everything as a string
- Encode everything as a number
- Use several columns for different (main) types

Sparql query language example

```
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
   ?country   a   type:LandlockedCountries ;
   rdfs:label   ?country_name ;
   prop:populationEstimate  ?population .
 FILTER
   (?population > 15000000 &&
     langMatches(lang(?country_name), "EN")) .
} ORDER BY DESC(?population)
```

RDFS

RDFS: RDF Schema

Three kinds of simple **taxonomy rules** added to RDF

„if X is a car, X is a vehicle“

„if X has a property profession, X is a person“

„if X has a property brother, value of the property is a person“

First (main) rule

example:

ex:MotorVehicle rdf:type rdfs:Class

exthings:companyCar rdf:type ex:Van

ex:Van rdfs:subClassOf ex:MotorVehicle

Main rule

same facts in the predicate calculus notation:

rdf:type(ex:MotorVehicle, rdfs:Class)

rdf:type(exthings:companyCar,ex:Van)

rdfs:subClassOf(ex:Van, ex:MotorVehicle)

built-in rule assumed in rdfs:

rdf:type(X,Y) & rdfs:subClassOf(Y,Z) => rdf:type(X,Z)

Second rule

ex:Person rdf:type rdfs:Class .
ex:isauthorof rdf:type rdf:Property .
ex:isauthorof rdfs:range ex:Person

and the facts

ex:person1 ex:isauthorof ex:hamlet

should derive:

ex:person1 rdf:type ex:Person

Logically ...

built-in rule assumed in rdfs:

Y(X,Z) & rdfs:range(Y,U) => rdf:type(X,U)

Third rule

ex:Book rdf:type rdfs:Class .
ex:isauthorof rdf:type rdf:Property .
ex:isauthorof rdfs:domain ex:Book .

and the fact

ex:person1 ex:isauthorof ex:hamlet

should derive:

ex:hamlet rdf:type ex:Book

Logically ...

built-in rule assumed in rdfs:

Y(X,Z) & rdfs:domain(Y,U) => rdf:type(Z,U)

Additional encoding layer!

built-in rules assumed in rdfs:

rdf:type(X,Y) & rdfs:subClassOf(Y,Z) => rdf:type(X,Z)
Y(X,Z) & rdfs:range(Y,U) => rdf:type(X,U)
Y(X,Z) & rdfs:domain(Y,U) => rdf:type(Z,U)

have to be encoded in classical 1st order logic as

rdf(X,rdf:type,Y) & rdf(Y,rdfs:subClassOf,Z) => rdf(X,rdf:type,Z)
rdf(X,Y,Z) & rdf(Y,rdfs:range,U) => rdf(X,rdf:type,U)
rdf(X,Y,Z) & rdf(Y,rdfs:domain,U) => rdf(Z,rdf:type,U)

Reification

Example:

Fact: **http://xx#121 ex:eesnimi "Jaan"**
Metainfo about the fact:
timestamp: 10jaan2008,
sisestaja:peeter,
usaldusvaarsus:0.9

Example encoded as triples (invent object "13"):

13 rdf:object http://xx#121
13 rdf:predicate ex:eesnimi
13 rdf:subject "Jaan"
13 ex:timestamp 10jaan2008
13 sisestaja peeter

Reification rule

built-in rule assumed in rdf:

rdf(X,rdf:subject,Y) &
rdf(X,rdf:predicate,Z) &
rdf(X,rdf:object,U)

<=>

rdf(Y,Z,U).

NOT provided in rdfs:

- cardinality constraints on properties, e.g., that a Person has exactly one biological father.
- specifying that a given property (such as ex:hasAncestor) is transitive, e.g., that if A ex:hasAncestor B, and B ex:hasAncestor C, then
  A ex:hasAncestor C.
- specifying that a given property is a unique identifier (or key) for instances of a particular class.
- specifying that two different classes (having different URIrefs) actually represent the same class.
- specifying that two different instances (having different URIrefs) actually represent the same individual.

NOT provided in rdfs:

- specifying constraints on the range or cardinality of a property that depend on the class of resource to which a property is applied, e.g., being able to say that

- for a soccer team the ex:hasPlayers property has 11 values,
- for a basketball team the ex:hasPlayers property has 5 values.

- the ability to describe new classes in terms of combinations (e.g., unions and intersections) of other classes, or to say that two classes are disjoint (i.e., that no resource is an instance of both classes).

Ways to extend rdfs

Two main approaches:

- Using traditional rules.
  - One of the popular options:
    RIF (Rule Interchange Format).
  - Older favorite:
    RuleML (Rule Meta-Language)
  - A complex exotic standard:
    CL (Common Logic)
- Using a specialised description-logic based language
  OWL (Web Ontology Language)

Semantic web

Started ca 2001. Promoted by Berners-Lee and W3C.

What is it?

Google:

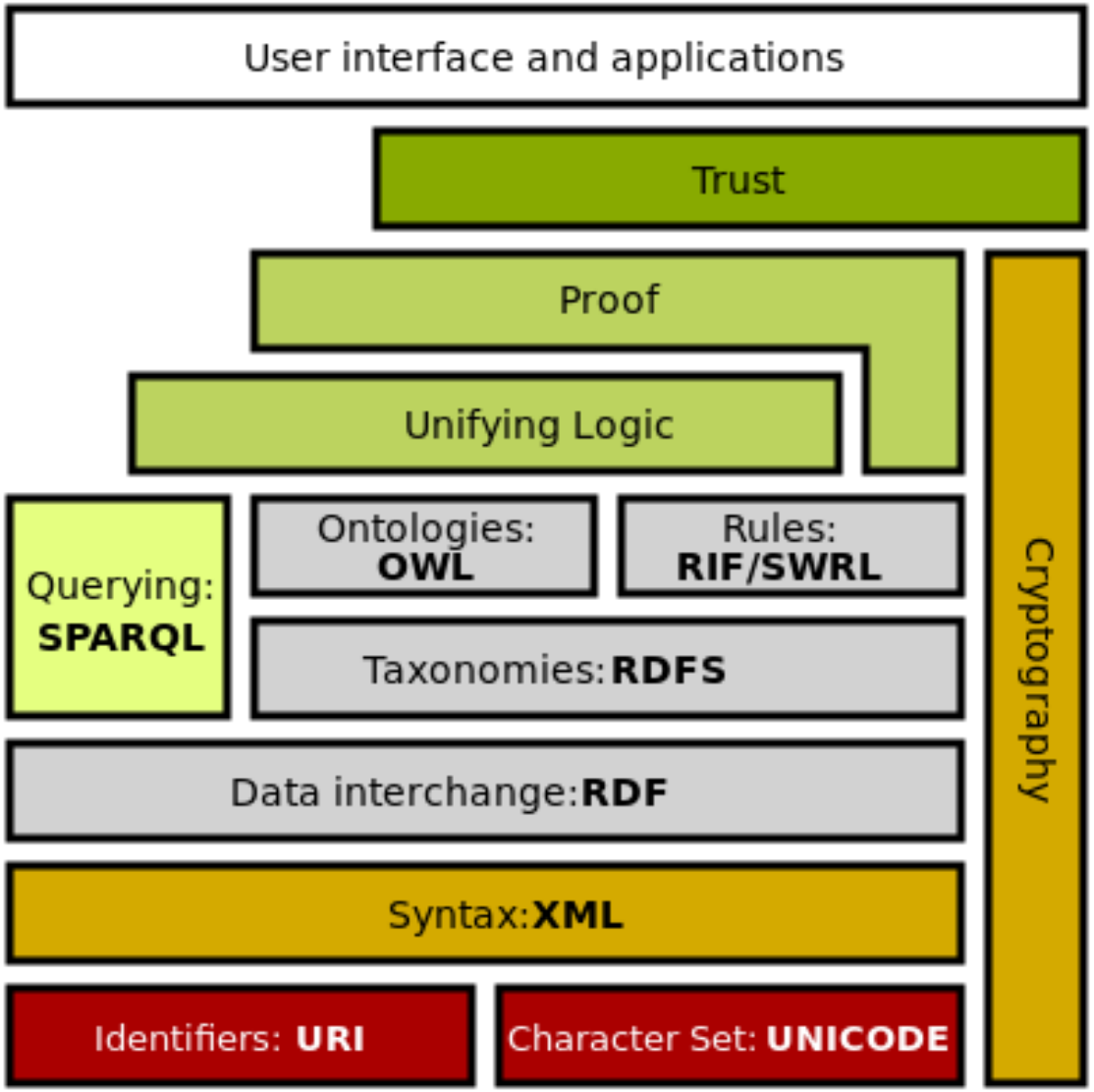
a proposed development of the World Wide Web in which data in web pages is structured and tagged in such a way that it can be read directly by computers.

"the Semantic Web could usher in a golden age of information access„

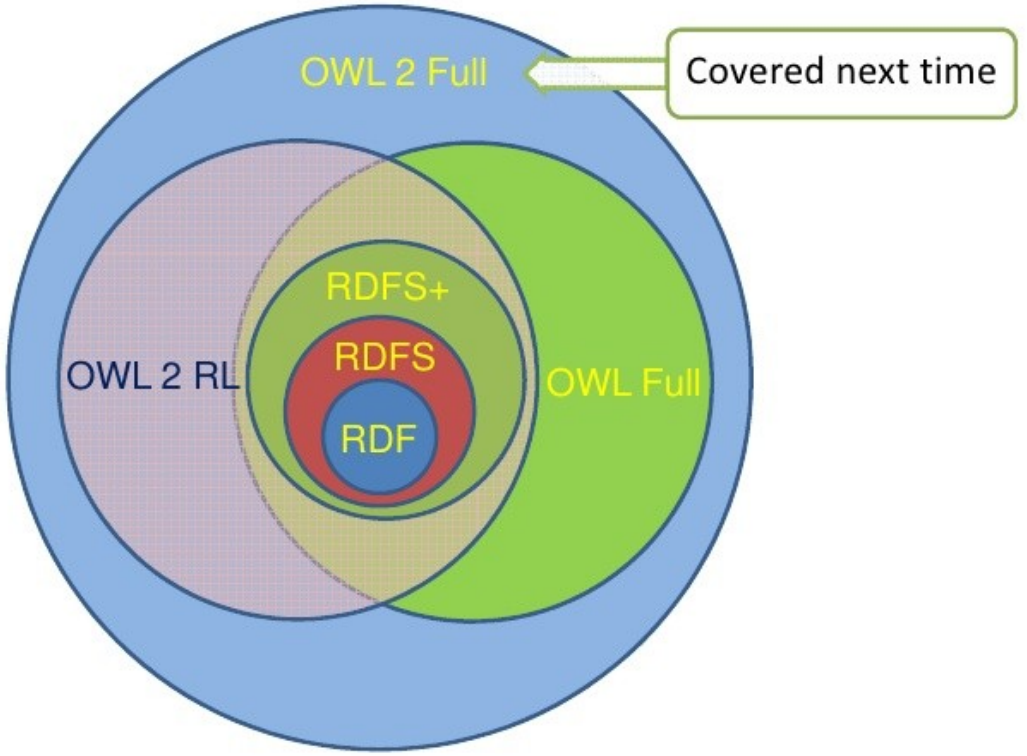
Wiki:

The Semantic Web is an extension of the World Wide Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

Semantic web hiearchy



OWL is a hierarchy of specialized logical languages on top of RDF and RDFS



Not much practical success, though

From a recent Sean Palmer lamenting email:

> In case you don't want to wade through the waffle, the summary is that there were roughly four phases of Semantic Web development starting with the eponymous golden age: Semantic Web (2001-2005), Linked Data (2006-2010), JSON-LD (2010-2014), and now the Data Activity (2015-).
>
> The biggest tangible results are Schema.org in conjunction with JSON-LD for SEO, Dbpedia, a few tacky database products, and very loosely the API economy.
>
> ....

Continued ...

From a recent Sean Palmer lamenting email:

> As I say, the Semantic Web was originally the conception of graphs instead of trees, with global symbols, published in an open and decentralised system.
>
> ...
>
> The main problem seems not to have been the proliferation of the Semantic junk such as RDF/XML and SPARQL, as is sometimes argued (references below), but rather that the Web side did not provide an open and decentralised system in which to host the Semantic side.
>
> Ongoing efforts are being made to rectify that, but there are no promising solutions in that domain and so many of the Semantic Web ideas will remain dormant.